Research Report 276

Agency and Autonomy: A Formal Framework

Michael Luck and Mark d'Inverno

RR276

With the recent rapid growth of interest in Multi-Agent Systems, both in artificial intelligence and software engineering has come an associated difficulty concerning basic terms and concepts. In particular, the terms *agency* and *autonomy* are used with increasing frequency to denote different notions with different connotations. In this paper we lay the foundations for a principled theory of agency and autonomy, and specify the relationship between them. Using the Z specification language, we describe a three-tiered hierarchy comprising objects, agents and autonomous agents where agents are viewed as objects with *goals*, and autonomous agents are agents with *motivations*.

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom

December 1994

Agency and Autonomy: A Formal Framework

Michael Luck
Dept. of Computer Science
University of Warwick
Coventry, CV4 7AL
United Kingdom
EMAIL mikeluck@dcs.warwick.ac.uk
TEL +44 1203 523193
FAX +44 1203 525714

Mark d'Inverno
School of Computer Science
University of Westminster
New Cavendish Street, London
United Kingdom
EMAIL dinverm@westminster.ac.uk
TEL +44 171 911 5000

Abstract

With the recent rapid growth of interest in Multi-Agent Systems, both in artificial intelligence and software engineering has come an associated difficulty concerning basic terms and concepts. In particular, the terms *agency* and *autonomy* are used with increasing frequency to denote different notions with different connotations. In this paper we lay the foundations for a principled theory of agency and autonomy, and specify the relationship between them. Using the Z specification language, we describe a three-tiered hierarchy comprising objects, agents and autonomous agents where agents are viewed as objects with *goals*, and autonomous agents are agents with *motivations*.

1 Introduction

1.1 Background

Although there has been substantial progress in artificial intelligence for many years, research tended to focus on solving problems without regard to any real external environment or to the notion of a reasoning *agent*. In other words, the problems and their solutions, while significant, were limited in that they were divorced from real situations. More recently, however, the importance of these limitations has been recognised. One consequence is the rapid growth of interest in the design and construction of *agents* as systems exhibiting intelligent behaviour.

Concepts of *agents* and *agency* are increasingly being used in a range of areas of artificial intelligence (AI) and computer science. However, these notions are often different, even within the same subfield, and the lack of a common understanding can stifle further research and development. In distributed AI, *agents* are often taken to be the same as *autonomous agents*, and the two terms are used interchangeably, without regard to their relevance or significance. The difference between these related but distinct notions is both important and useful in considering aspects of intelligence.

In this paper, we define *autonomy* and *agency*, and explicate the relationship between them. We argue that autonomy is distinct and is achieved by *motivating* agency. Our concern is to develop a formal model of agency and autonomy that can be used both as the basis of an implementation, and also as a precise but general framework for further research.

1.2 Benevolence and Autonomy

In traditional models of goal adoption, goals are broadcast by one agent, and adopted by other agents according to their own relevant competence [18]. This assumes that agents are already designed with common or non-conflicting goals that allow mutual assistance in satisfying additional goals. Negotiation as to how these additional goals are satisfied typically takes the form of mere goal-node allocation. Thus an agent simply has to communicate its goal to another agent, and cooperation will ensue. This concept of *benevolent agents* [11] who cooperate with other agents whenever and wherever possible, severely constrains the richness of behaviour that can be achieved by autonomous agents, by virtue of its simplistic and limited view of the world. An autonomous agent may exhibit benevolent behaviour, but a purely and exclusively benevolent agent is not autonomous [4]. Cooperation will occur between two agents only when it is considered advantageous to each. Autonomous agents are thus self-interested agents. In this view, autonomous benevolent behaviour is possible, but arises only through the self-interests of an agent. A goal (whether viewed as 'selfish' or 'altruistic') will always be adopted so as to satisfy a self-interested motivation [1].

As stated above, the terms *agent* and *agency* are used to denote different concepts. Given the range of areas in which the terms are applied, this lack of consensus over meaning is not surprising. As Shoham [13] points out, the number of diverse uses of the term *agent* are so many that it is almost meaningless without reference to a particular notion of agenthood.

In a recent collection of papers, for example, several different views emerge. For Smith [17], an agent is a "persistent software entity dedicated to a specific purpose." Selker [12] takes agents to be "computer programs that simulate a human relationship by doing something that another person could do for you." More loosely, Riecken [10] refers to "integrated reasoning processes" as agents. Others avoid the issue completely, and leave the interpretation of their agents to the reader. It is recognised, however, that there is no agreement on what it is that makes something an agent.

1.3 Agency, Autonomy and Formality

If we are to confront the problems that arise from this lack of agreement and definition, then the use of formalisms is appropriate since it allows unambiguous descriptions of complex systems. Formal specification languages not only provide a means for unambiguous specification but also provide proof systems and sets of proof obligations which enable the construction of reliable and robust models. Typically, such languages are based on set theory and first order logic. In the current work, we have adopted the specification language Z [19] for two major reasons. First, it provides modularity and abstraction and is sufficiently expressive to allow a consistent, unified and structured account of a computer system and its associated operations. Such structured specifications enable the description of systems at different levels of abstraction, with system complexity being added at successively lower levels. Second, we view our enterprise as that of building programs. Z schemas are particularly suitable in squaring the demands of formal modelling with the need for implementation by allowing transition between specification and program. Thus our approach to formal specification is pragmatic — we need to be formal to be precise about the concepts we discuss, yet we want to remain directly connected to issues of implementation. Z provides just those qualities that are needed, and is increasingly being used for specifying frameworks and systems in AI (e.g.

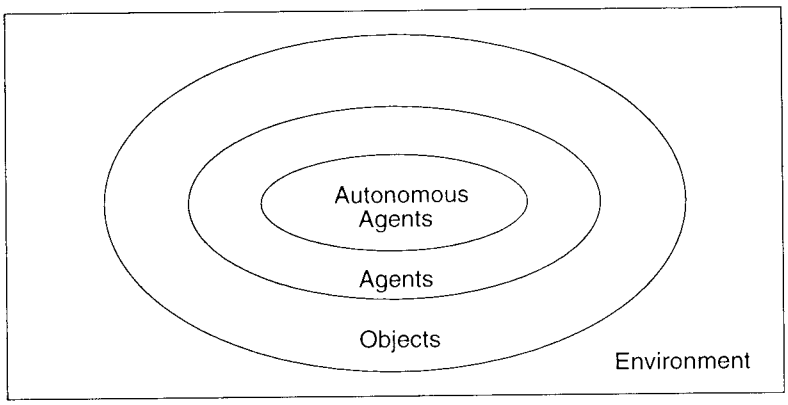

Figure 1: The Entity Hierarchy

[7, 5, 3]).

2 Initial Concepts

2.1 Introduction

It was stated earlier that there is a lack of consensus over the meaning of the term agent, and that there are many diverse notions of agency. In this section, we introduce terms and concepts that will be used to explicate our understanding of *agents* and *agency*, and provide formal definitions that are used subsequently in developing the concept of *autonomy*.

Our view of agency best relates to that of Shoham [13] who takes an *agent* to be any entity to which mental state can be ascribed. According to Shoham, such mental state consists of components such as beliefs, capabilities and commitments, but there is no unique correct selection of them. This is sensible, and we too do not demand that all agents necessarily have the same set of mental components. Indeed, we recognise the limitations associated with assuming an environment comprising homogeneous agents, and consequently include in this discussion heterogeneous agents with varying capabilities. We do, however, specify what is minimally *required* of an entity for it to be considered an agent.

The definition of agency that follows is intended to subsume existing concepts as far as possible. It will be built up and extended over the course of the paper, and will involve several intermediate concepts. In short, we propose a three-tier hierarchy of entities comprising *objects*, *agents* and *autonomous agents*. The basic idea underlying this hierarchy is that all known entities are objects. Of this set of objects, some are agents, and of these agents some are autonomous agents. This is shown as a Venn diagram in Figure 1. The following sections define what is required for each of the entities in the hierarchy.

Before we can move to a definition of any of these entities, we must first define some primitives. The first of these is an *action* which is strongly related to the notion of *agency*.

Definition: An *action* is a discrete event which changes the state of the environment.

It is certainly possible that actions may not be atomic, but for the purposes of our model, it is assumed that the property of atomicity holds for actions without any loss of generality.

The second primitive that needs defining is an *attribute*. Attributes are simply features of the world, and are the only characteristics which are manifest. They need not be perceived by any particular entity, but must be potentially perceivable in an omniscient sense. (The notion of a feature here allows anything to be included.)

Definition: An *attribute* is a perceivable feature.

Using the notions of actions and attributes, we provide definitions of agency and autonomy. In order to do so, we must distinguish agents and autonomous agents from other entities. We call these other entities *objects*, and discuss them below.

2.2 Object Specification

At a basic level, an object can be defined in terms of its abilities and its attributes. An object, in this sense, is just a 'thing' or entity with no further defining characteristics. This provides us with the basic building block to develop our notion of agency.

Definition: An *object* comprises a set of actions and a set of attributes.

In Z, before constructing a specification, we must first define types. Here we define the set of all actions and the set of all attributes:

$$[Action, Attribute]$$

Now a **state schema** can be constructed that defines an object. Z schemas have two parts: the upper, **declarative**, part which declares variables and their types, and the lower, **predicate**, part which relates and constrains those variables. The type of any schema can be considered as the cartesian product of the types of each of its variables, without any notion of order, but constrained by predicates.

The schema below has only a declarative part containing two variables. First, *capableof* is the set of actions of the object, and is sometimes referred to as the *competence* of the object. Second, *attributes* is the set of features of the object. Objects are therefore defined by their ability in terms of their actions, and their configuration (in a broad sense) in terms of their attributes. The configuration of an object includes references to the body of the object and its position, and is similar to the notion used by Goodwin [5].

Object

$$capableof : \mathbb{P}\, Action$$
$$attributes : \mathbb{P}\, Attribute$$

Attributes are manifest and are potentially accessible by an observer. The capabilities of an object in terms of actions, by contrast, are not necessarily observable.

For example, consider a robot without a power supply. Since the robot has no power, its capabilities are severely limited and include just those which rely on its physical presence, such as supporting things, weighing things down, and so on. The attributes of the robot specify that it is blue, that it is upright, that it is large, and other such features. As a second example, consider a coffee-cup. Its capabilities state that it can support things and that it can contain liquid, for example, while its attributes state that it is stable, white and made of china. (These examples will be developed further to distinguish between agents and autonomous agents.)

An object must be situated in an environment. We take an environment simply to be a set of attributes that describes all of the features of the world and all of the entities (objects, agents or autonomous agents) in the world. (There is a circularity here which will be discussed later.)

$$Environment == \mathbb{P}\ Attribute$$

We must also specify an action-selection function, which determines the next actions of an object in an environment. For example, if a piece of paper is placed on top of our robot, then we can say that the robot will *support* the paper. Or, if tea is poured into the coffee-cup, then the cup will *contain* the tea. In both of these cases, it is the environment that determines the next action of the object through the placement of the paper and the pouring of the tea. The actions are thus *passive* actions.

In the *ObjectAct* schema below, the definitions from the *Object* schema are included, and then the action-selection function, *objectactions*, is defined as a total function from *Environment* to $\mathbb{P}\ Action$. That is, given any environment it will return a (possibly empty) set of actions. The assertion in the predicate part of the schema constrains the next actions to be taken (given by applying *objectactions*) to be within the object's competence. It can be read precisely as: given any environment, applying *objectactions* to that environment will return a set of actions which is a subset of the agents capabilities.

ObjectAct

$$Object$$
$$objectactions : Environment \rightarrow \mathbb{P}\ Action$$

$$\forall\ env : Environment \bullet (objectactions\ env) \subseteq capableof$$

We can also declare an instance of an environment:

Env

$$environment : Environment$$

$$environment \neq \{\}$$

Now, we can define the state of an object in its environment. In the *ObjectState* schema, the information from the *Env* and *ObjectAct* schemas is included, along with *history* which represents the sequence of (sets of) actions which the object has performed. Each of these sets of actions must comprise only those actions of which the object is capable, which is stated in the schema predicate as: any set in the range of this sequence must be a subset of *capableof*. The predicate is meaningful because sequences are defined in Z to be functions from the set of natural numbers to the components of the sequence, and so the range of a sequence is just the set containing the elements of the sequence.

The attributes of the object are also constrained to be a subset of those attributes in the environment. This is because the object is part of the environment and its attributes must therefore also be attributes of the environment.

To aid schema readability, we can also add redundant or *recoverable* information to the schema. For example, the variable, *willdo*, specifies the next actions that the agent will do, which is redundant since it is recoverable in exactly the way it is specified. It is determined by applying

the *objectactions* function to the *environment*. This recoverable information is separated from the rest of the schema by use of a small space. In what follows, we use this convention to add such additional information to a schema.

ObjectState

$$
\begin{array}{l}
Env \\
ObjectAct \\
history : \mathsf{seq}\,(\mathbb{P}\,Action) \\
\\
willdo : \mathbb{P}\,Action \\
\hline
\forall\, as : \mathsf{ran}\, history \bullet as \subseteq capableof \\
\\
willdo = objectactions\; environment
\end{array}
$$

These definitions have an inherent circularity, however. Objects are situated in environments, but environments are just sets of attributes derived from the entities (or objects) in the environment. The attributes of the environment must therefore include those of the object itself. This is a useful abstraction mechanism that serves to reduce the complexity of the world until we need richer information.

Everything in the environment is represented simply in terms of attributes. If the environment is always accessible, then we need only note those attributes that are relevant to the task at hand, and can limit the information flow. With resource bounds, such restrictions are critical. Whenever it is necessary to access information about the environment, we instantiate object models (or schemas) with relevant attributes, and ignore the remainder. Thus we distinguish objects from the environment (by clustering attributes, for example,) only when it is useful.

This notion of instantiation is the basis of this framework, for through successive instantiation we increase the complexity of entities to introduce new concepts. Moreover, it is a notion that is effective in focusing attention on relevant aspects, and limiting the amount of effort needed to function capably in real-world situations.

2.3 Object Operations

So far, we have described an object and the way in which its next actions are selected. Next we describe how the performance of these actions affects the environment in which the object is situated. To do this, we need to introduce **operation schemas**. Operation schemas in Z describe the relationship between two states; the state before the operation, and the state after the operation. After-state variables are distinguished with dashes.

First, we define which variables of a state remain unchanged after a set of actions. The $\Delta ObjectState$ schema states that when an *ObjectState* changes (through performing actions), its competence does not change, and its action selection function does not change. (The Δ is the naming convention to describe a change of state.)

$\Delta ObjectState$

$ObjectState$

$ObjectState'$

$capableof' = capableof$

$objectactions' = objectactions$

An *interaction* is simply what happens when actions are performed in an environment. The effects of an interaction on the environment are determined by applying the *effectinteraction* function in the axiom definition below to the current environment and the actions taken. This axiom definition is a global variable and is consequently always in scope. We require only one function to describe all interactions, since an action will result in the same change to an environment whether taken by an object, agent or autonomous agent.

$$effectinteraction : Environment \longrightarrow \mathbb{P}\, Action \nrightarrow Environment$$

When an object interacts in its environment and performs an action, both the state of the object and the environment change, represented in *ObjectEnvInteract*, by $\Delta ObjectState$. (Remember that *ObjectState* includes *Env*.) The history variable is updated by concatenating the current set of actions to the end of the sequence, and the new environment is given by applying *effectinteraction* to the old environment and the current actions. The last line states that the actions which follow the current actions are found by applying *objectactions* to the new environment.

$ObjectEnvInteract$

$\Delta ObjectState$

$history' = history \frown \langle willdo \rangle$

$environment' = effectinteraction\ environment\ willdo$

$willdo' = objectactions\ environment'$

The characterizing features are the set of actions which can be performed by the object and the attributes of the object. This is not sufficient for agency, and without further detail or specialization, is inadequate for a definition of agency.

3 Agency

3.1 Introduction

There are many dictionary definitions for an agent. A recent paper by Wooldridge and Jennings [20] quotes the definition of an agent as "one who, or that which, exerts power or produces an effect."[1] However, they omitted the second sense of agent which is given as "one who acts for another..." This is important, for it is not the acting alone that defines agency, but the acting for *someone or something* that is defining. Indeed, Wooldridge and Jennings acknowledge the difficulties in a purely action-based analysis of agency.

[1] *The Concise Oxford Dictionary of Current English (7th edition)*, Oxford University Press.

To understand better what actually constitutes an agent, consider the example of a coffee-cup. A cup is an object. We can regard it as an agent and ascribe to it mental state, but it serves no useful purpose to do so without considering the circumstances. A cup is an agent *if* it is containing a liquid and it is doing so to some end. In other words, if I fill a cup with coffee, then the cup is my agent — it serves my purpose. Alternatively, the cup would also be an agent if it was placed upside down on a stack of papers and used as a paperweight. It would *not* be an agent if it was just sitting on a table without serving any purpose to any agent. In this case it would be an object. Note that we do not require an entity to be intelligent for it to be an agent.

Thus agents are just objects with certain dispositions. They may always be agents, or they may revert to being objects in certain circumstances. This will be explored further later. For the moment we will concentrate on the nature of the disposition that characterizes an agent.

An object is an agent if it serves a useful purpose either to a different agent, or to itself, in which case the agent is *autonomous*. This latter case is discussed further later. Specifically, an agent is something that 'adopts' or satisfies a goal or set of goals (often of another). Thus if I want to store coffee in a cup, then the cup is my agent for storing coffee. It has been *ascribed* or has *adopted* my goal to have the coffee stored. An agent is thus defined in relation to its goals. We take a traditional view of goals as describable environmental states.

Definition: A *goal* is a state of affairs to be achieved in the environment.

Definition: An *agent* is an instantiation of an object together with an associated goal or set of goals.

3.2 Agent Specification

Before we can define a schema for agency, we must define goals to be just a set of attributes that describe a state of affairs in the world:

$$Goal == \mathbb{P}\ Attribute$$

The schema for an agent is simply that of an object but with the addition of goals.

Agent

$$Object$$
$$goals : \mathbb{P}\ Goal$$

$$goals \neq \{\}$$

Thus an agent has or is *ascribed* a set of goals which it retains over any instantiation (or lifetime). One object may give rise to different instantiations of agents. An agent is instantiated from an object in response to another agent. Thus agency is *transient*, and an object which becomes an agent at some time may subsequently revert to being an object.

Returning to the cup example, we have an agent with the same attributes and actions as the cup object, but now it can be ascribed the goal — my goal — of *storing my coffee*. Not everyone will know that it is an agent in this way, however. If, for example, I am in a cafe and there is a half-full cup of lemon-tea on my table, there are several views that can be taken. It can be regarded by the waiter as an agent for me, storing my tea, or it can be regarded as

an object serving no purpose if the waiter thinks it is not mine. The view of the cup as an object or agent is relevant to whether the waiter will remove the cup or leave it at the table. Note that we are not suggesting that the cup actually possesses a goal, just that there is a goal that it is satisfying.

Consider also the robot example, and suppose now that the robot has a power supply. If the robot has no goal, then it cannot use its actuators in any sensible way but only, perhaps, in a random way, and must be considered an object. Alternatively, if the robot has some goal or set of goals, which allow it to employ its actuators in some directed way, such as picking up a cup, or carrying a table, or riveting a panel onto a hull, then it is an agent. The goal need not be explicitly represented, but can instead be implicit in the hardware or software design of the robot. It is merely necessary for there to be a goal of some kind. (We also allow goals in agents to be replaced by higher level motivations, but discuss that later in the context of autonomy.)

These examples highlight the range of behaviour that is available from agents. The coffee-cup is passive and has goals *imposed* upon and *ascribed* to it, while the robot is capable of actively manipulating the environment by performing actions designed to satisfy its goals.

We now introduce perception. An agent in an environment may have a set of percepts available. These are the possible attributes that an agent could perceive subject to its capabilities and current state. However, due to limited resources, an agent will not normally be able to perceive all those attributes possible, and bases action on a subset, which we call the *actual* percepts of an agent. Some agents will not be able to perceive at all. In the case of a cup, for example, the set of possible percepts will be empty, and consequently the set of actual percepts will also be empty. The robot, however, may have several sensors which allow it to perceive. Thus it is not a requirement of an agent that it is able to perceive.

For clarity of exposition, we define a *View* to be the perception of an *Environment* by an agent, and which is just a set of attributes.

$$View == \mathbb{P}\, Attribute$$

It is also important to note that it is only meaningful in our model to consider perceptual abilities in the context of goals (or motivations which give rise to goals). Thus, when considering objects which have no goals, perceptual abilities are not relevant. Objects respond directly to their environments and make no use of percepts even if they are available. We say that perceptual capabilities are *inert* in the context of objects.

In the schema for agent perception, *AgentPercepts*, we add further detail to the definition of agency, and so include the schema *Agent*. An agent has a set of perceiving actions which are a subset of the capabilities of an agent. The function, *canperceive*, determines the attributes that are potentially available to an agent through its perception capabilities. When applied, its arguments are the current environment and the agent's capabilities. The second predicate line states that those capabilities will be precisely the set of perceptual capabilities. Finally, the function, *willperceive*, describes those attributes which are actually perceived by an agent and will always be applied to its goals.

AgentPercepts

$$
\begin{array}{l}
Agent \\
perceivingactions : \mathbb{P}\ Action \\
canperceive : Environment \longrightarrow \mathbb{P}\ Action \nrightarrow Environment \\
willperceive : \mathbb{P}\ Goal \longrightarrow Environment \longrightarrow View \\
\hline
perceivingactions \subseteq capableof \\
\forall\ env : Environment;\ as : \mathbb{P}\ Action \bullet \\
\qquad as \in \mathsf{dom}\,(canperceive\ env) \Rightarrow as = perceivingactions \\
\mathsf{dom}\ willperceive = \{goals\}
\end{array}
$$

Directly corresponding to the goal or goals of an agent, is an action-selection function, dependent on the goals, current environment and the actual perceptions. This is specified in *AgentAct* below, in the same way as *ObjectAct* previously, the related predicate ensuring that the function returns a set of actions within the agent's competence. Note also that if there are no perceptions, then the action-selection function is dependent only on the environment, as it is with *objectactions* in *ObjectAct*.

AgentAct

$$
\begin{array}{l}
Agent \\
agentactions : \mathbb{P}\ Goal \longrightarrow View \longrightarrow Environment \longrightarrow \mathbb{P}\ Action \\
\hline
\forall\ gs : \mathbb{P}\ Goal;\ v : View;\ env : Environment \bullet (agentactions\ gs\ v\ env) \subseteq capableof \\
\mathsf{dom}\ agentactions = \{goals\}
\end{array}
$$

We also define the state of an agent. This includes two variables, *posspercepts*, describing those percepts possible in the current environment, and *actualpercepts*, a subset of these which are the current (actual) percepts of the agent in the current environment. These variables are calculated using the *canperceive* and *willperceive* functions respectively. Notice that the *objectactions* variable from the *ObjectAct* schema is related to the *agentactions* variable from the *AgentAct* schema. Since goals are fixed, changes to the *actualpercepts* of the agent affect the functionality of the agent when compared to the base object from which it is created.

AgentState

$$
\begin{array}{l}
AgentPercepts \\
AgentAct \\
ObjectState \\
posspercepts : Environment \\
actualpercepts : View \\
\hline
actualpercepts \subseteq posspercepts \\
posspercepts = canperceive\ environment\ perceivingactions \\
actualpercepts = willperceive\ goals\ posspercepts \\
objectactions = agentactions\ goals\ actualpercepts \\
\\
(perceivingactions = \{\}) \Rightarrow (posspercepts = \{\}) \\
willdo = agentactions\ goals\ actualpercepts\ environment
\end{array}
$$

3.3 Agent Operations

As with objects, we define which of the agent state variables remain unchanged after a set of actions has been performed by that agent. If any of these variables ever did change, a different agent schema would have to be instantiated.

$\Delta AgentState$

$AgentState$
$AgentState'$

$capableof' = capableof$
$goals' = goals$
$perceivingactions' = perceivingactions$
$canperceive' = canperceive$
$willperceive' = willperceive$
$agentactions' = agentactions$

We now specify how an agent interacts with its environment. As a result of an interaction, the environment changes and agent state changes with certain variables unaffected as defined in $\Delta AgentState$. The history and environment are altered in exactly the same way as was described in the equivalent *Object* schema, and the final four predicates of *AgentEnvInteract* are redundant, but show explicitly how the schema variables are updated.

$AgentEnvInteract$

$\Delta AgentState$

$history' = history \frown \langle willdo \rangle$
$environment' = effectinteraction\ environment\ willdo$

$posspercepts' = canperceive\ environment'\ perceivingactions$
$actualpercepts' = willperceive\ goals\ posspercepts'$
$objectactions' = agentactions\ goals\ actualpercepts'$
$willdo' = agentactions\ goals\ actualpercepts'\ environment'$

4 Autonomy as Motivated Agency

4.1 Introduction

So far we have developed a definition of agency. However, the definition relies upon the existence of other agents which provide goals that are adopted in order to instantiate an agent. In order to ground the chain of goal adoption, to escape what could be an infinite regress, and also to bring out the notion of *autonomy*, we introduce *motivation*.

Grounding the hierarchies of goal adoption demands that we have some agents which can generate their own goals. These agents are *autonomous* agents since they are not dependent on the goals of others. Autonomous agents possess goals which are *generated* from within rather than *adopted* from other agents. These goals are generated from *motivations*, which can be considered to be higher-level non-derivative goals which characterize the nature of

the agent. Motivations are, however, qualitatively different in that they are not describable states of affairs in the environment. For example, consider the motivation *greed*. This does not specify a state of affairs to be achieved, nor is it describable in terms of the environment, but it may (if other motivations permit) give rise to the generation of a goal to rob a bank. The distinction between the motivation of greed and the goal of robbing a bank is clear, with the former providing a reason to do the latter, and the latter specifying what must be done.

Definition: A *motivation* is any desire or preference that can lead to the generation and adoption of goals and which affects the outcome of the reasoning or behavioural task intended to satisfy those goals.

(This draws on the definition used by Kunda [6].)

A *motivated agent* is thus an agent that pursues its own agenda for reasoning and behaviour in accordance with its internal motivation. Since motivations ground the goal-generation regress, we claim that it is motivation that is the critical factor in achieving autonomy. An *autonomous agent* must necessarily be a *motivated* agent.

Definition: An *autonomous agent* is an instantiation of an agent together with an associated set of motivations.

4.2 Autonomous Agent Specification

We can now specify an autonomous agent. First we define the set of all motivations as a base type:

$$[Motivation]$$

An autonomous agent is defined as an agent with motivations and some potential means of evaluating behaviour in terms of the environment and these motivations. In other words, the behaviour of the agent is determined by both external and internal factors. This is qualitatively different from an agent with goals because motivations are non-derivative and governed by internal inaccessible rules, while goals are derivative and relate directly to motivations.

AutonomousAgent

$$Agent$$
$$motivations : \mathbb{P}\ Motivation$$

$$motivations \neq \{\}$$

In illustration of these ideas, note that the cup cannot be considered autonomous because it cannot generate its own goals. The robot, however, is potentially autonomous in the sense that it may have a mechanism for internal goal generation depending on its environment. Suppose the robot has motivations of achievement, hunger and self-preservation, where achievement is defined in terms of fixing tyres onto a car on a production line, hunger is defined in terms of maintaining power levels, and self-preservation is defined in terms of avoiding system breakdowns. In normal operation, the robot will generate goals to attach tyres to cars through a series of subgoals. If its power levels are low, however, it may replace the goal of attaching tyres with a newly-generated goal of recharging its batteries. A third possibility is that in satisfying its achievement motivation, it works for too long and is in danger of overheating. In this case, the robot can generate a goal of pausing for an appropriate period in order to

avoid any damage to its components. Such a robot is autonomous because its goals are not imposed, but are generated in response to its environment.

Autonomous agents also perceive, but motivations, as well as goals, filter relevant aspects of the environment. In the schema below, the function *autowillperceive* is then a more complex version of an agent's *willperceive*, but they are related – and must be since an autonomous agent is still an agent – as shown in the schema. However, that which an autonomous agent is *capable* of perceiving at any time is independent of its motivations. Indeed, it will always be independent of goals and motivations, and there is consequently no equivalent increase in functionality to *canperceive*.

AutonomousAgentPercepts

$$\begin{array}{l} AutonomousAgent \\ AgentPercepts \\ autowillperceive : \mathbb{P}\, Motivation \longrightarrow \mathbb{P}\, Goal \longrightarrow Environment \longrightarrow View \\ \hline willperceive = autowillperceive\ motivations \\ \mathsf{dom}\ autowillperceive = \{motivations\} \end{array}$$

The next schema defines the action-selection function and includes the previous schema definitions for *AgentAct* and *AutonomousAgent*. The action-selection function for an autonomous agent is produced at every instance by the motivations of the agent, and is always and only ever applied to the motivations of the autonomous agent.

AutonomousAgentAct

$$\begin{array}{l} AutonomousAgent \\ AgentAct \\ autoactions : \mathbb{P}\, Motivation \rightarrow \mathbb{P}\, Goal \rightarrow View \rightarrow Environment \rightarrow \mathbb{P}\, Action \\ \hline \mathsf{dom}\ autoactions = \{motivations\} \\ agentactions = autoactions\ motivations \end{array}$$

Next, we define the state of an autonomous agent in an environment, and include the *AgentState*, *AutonomousAgentPercepts* and *AutonomousAgentAct* schemas.

AutonomousAgentState

$$\begin{array}{l} AutonomousAgentPercepts \\ AutonomousAgentAct \\ AgentState \\ \hline willdo = autoactions\ motivations\ goals\ actualpercepts\ environment \end{array}$$

4.3 Autonomous Agent Operations

As with objects and agents, we define which of the variables that characterize the autonomous agent state remain unchanged after a set of actions has been performed by that agent.

$\Delta AutonomousAgentState$

$AutonomousAgentState$
$AutonomousAgentState'$

$capableof' = capableof$
$perceivingactions' = perceivingactions$
$canperceive' = canperceive$
$autowillperceive' = autowillperceive$
$autoactions' = autoactions$

Now we specify the operation of an autonomous agent performing its next set of actions in its current environment. Notice that while no explicit mention is made of any change in motivations, they may change in response to changes in the environment. If they do change, then the agent functions *willperceive* and *agentactions* will also change. Further, motivations may generate new and different goals for the agent to pursue. In any of these cases, the characterizing features of an agent are in flux so that an autonomous agent can be regarded as a continually re-instantiated non-autonomous agent. In this sense, autonomous agents are permanent as opposed to transient non-autonomous agents (which may revert to being objects).

$AutonomousAgentEnvInteract$

ΔEnv
$\Delta AutonomousAgentState$

$history' = history \frown \langle willdo \rangle$
$environment' = effectinteraction\ environment\ willdo$
$willperceive' = autowillperceive\ motivations'$
$agentactions' = autoactions\ motivations'$

$posspercepts' = canperceive\ environment'\ perceivingactions$
$actualpercepts' = willperceive'\ goals'\ posspercepts'$
$objectactions' = agentactions'\ goals'\ actualpercepts'$
$willdo' = autoactions\ motivations'\ goals'\ actualpercepts'\ environment'$

5 Discussion

There exists a small body of work that provides a similar view to that presented here. For example, Covrigaru and Lindsay [2] describe a set of properties that *characterize* autonomous systems to some "degree", relating to such factors as type and number of goals, complexity, interaction, robustness, and so on. Our work, in contrast, takes an absolute stand on autonomy in that it either exists or does not. Moreover, we *define* what is necessary for a system to be autonomous in very precise terms, and we distinguish clearly between objectness, agency and autonomy. One particular consequence of the difference in views is that we allow a rock, for example, to be considered an agent *if* it is being used for some purpose, such as a hammer for tent-pegs. Covrigaru and Lindsay deny the rock the quality of autonomy because it is not goal-directed, but ignore the possibility of agency, skipping over an important part of our framework.

The notion of motivation is not new, and has been used elsewhere. Simon, for example, takes motivation to be "that which controls attention at any given time," and explores the relation of motivation to information-processing behaviour, but from a cognitive perspective [14]. More recently, Sloman [16, 15] has elaborated on Simon's work, showing how motivations are relevant to emotions and the development of a computational theory of mind. Others have used motivation and related notions in developing computational architectures for autonomous agents such as the *motives* of Norman and Long [9], and the *concerns* of Moffat and Frijda [8]. What is new about the current work is the role of motivation in defining autonomy.

6 Conclusions

In the previous sections, we have constructed a formal specification which identifies and characterizes those entities that are called agents and autonomous agents. The work is not based on any existing classifications or notions because there is no consensus. Recent papers define agents in wildly different ways if at all, and this makes it extremely difficult to be explicit about their nature and functionality. The taxonomy given here serves several purposes:

- It provides clear and precise definitions for objects, agents and autonomous agents that allow a better understanding of the functionality of different systems.
- It explicates those factors that are necessary for agency and autonomy.
- It is sufficiently abstract to cover the gamut of agents, hardware and software, intelligent and unintelligent, etc.
- It allows further levels of specification to be added to describe particular agent designs and architectures.

In summary, we have defined and specified three classes of entity, *objects*, *agents* and *autonomous agents*. The distinctions between them are simple but strong and clear.

The framework provides an important basis for reference. We can classify both human and artificial agents equally well. Consider the relationship of a programmer to a program. Programs are always designed to satisfy goals, but these goals are rarely explicit or able to be modified independently of the programmer. The programs lack goal-generating motivations, but can be ascribed goals. In this respect, they are agents. Programmers typically develop programs according to several motivations which determine how the program is constructed. Time and effort must be balanced against cost, ease of use, simplicity, functionality and other factors. Programmers consider these factors in determining the design of programs and in the goal or goals of programs. Programmers can change the goals of programs by modifying code if desired, and can modify their own goals to suit circumstances. In this respect, programmers are autonomous agents.

The difference between these kinds of programs as agents and much recent use of the term is that the relationship between the user (or programmer) and the program has become explicit. Software agents assist users. They adopt the goals of the users in the tasks that they perform. Whether or not they are autonomous depends on the ability of the agents to function independently of those users, and to modify their goals in relation to circumstances. This paper has made explicit the relationships that have previously been implicit across the

board in the vast range of work on agency and autonomy, and provides a strong formal base on which to build.

One aspect not addressed here, however, is exactly *how* goals are generated from motivations. This is a difficult issue, but work is progressing on developing goal-generation mechanisms, including efforts described in the related work section above. The next stage of this work is to formalize a model of goal generation in the same way as the framework has been constructed here, developing a complete architecture for autonomous agents.

References

[1] R. Conte, M. Miceli, and C. Castelfranchi. Limits and levels of cooperation: Disentangling various types of prosocial interaction. In Y. Demazeau and J. P. Mueller, editors, *Decentralized AI*, volume II, pages 147–157, Amsterdam, 1991. Elsevier.

[2] A. A. Covrigaru and R. K. Lindsay. Deterministic autonomous systems. *AI Magazine*, 12(3):110–117, 1991.

[3] I. D. Craig. The formal specification of ELEKTRA. Research Report RR261, Department of Computer Science, University of Warwick, 1994.

[4] J. R. Galliers. A strategic framework for multi-agent cooperative dialogue. In *Proceedings of the 8th European Conference on Artificial Intelligence*, pages 415–420, 1988.

[5] R. Goodwin. Formalizing properties of agents. Technical Report CMU-CS-93-159, Carnegie-Mellon University, 1993.

[6] Z. Kunda. The case for motivated reasoning. *Psychological Bulletin*, 108(3):480–498, 1990.

[7] B. G. Milnes. A specification of the Soar architecture in Z. Technical Report CMU-CS-92-169, School of Computer Science, Carnegie Mellon University, 1992.

[8] D. Moffat and N. H. Frijda. An agent architecture: Will. In *Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages*, 1994.

[9] T. J. Norman and D. Long. A proposal for goal creation in motivated agents. In *Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages*, 1994.

[10] D. Riecken. An architecture of integrated agents. *Communications of the ACM*, 37(7):107–116, 1994.

[11] J. S. Rosenchien and M. R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 91–99, 1985.

[12] T. Selker. A teaching agent that learns. *Communications of the ACM*, 37(7):92–99, 1994.

[13] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.

[14] H. A. Simon. Motivational and emotional controls of cognition. In *Models of Thought*. pages 29 38. Yale University Press, 1979.

[15] A. Sloman. Motives, mechanisms, and emotions. *Cognition and Emotion*, 1(3):217–233. 1987.

[16] A. Sloman and M. Croucher. Why robots will have emotions. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*. pages 197 202, Vancouver. B.C., 1981.

[17] D. C. Smith, A. Cypher, and J. Spohrer. Programming agents without a programming language. *Communications of the ACM*, 37(7):55–67, 1994.

[18] R. G. Smith and R. Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man and Cybernetics*. 11(1):61–70, 1981.

[19] J. M. Spivey. *The Z Notation*. Prentice Hall, Hemel Hempstead, 2nd edition. 1992.

[20] M. J. Wooldridge and N. R. Jennings. Agent theories, architectures, and languages: A survey. In *Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages*. 1994.